REMARKS

Claims 1, 4-8, 11-15, 18-19, 22-28, 30-32, 34-36 and 38-39 are pending in the present application. Applicants respectfully request reconsideration of the application in view of the amendments and remarks made herein.

I. Rejections Under 35 U.S.C. § 103

Claims 1, 4-8, 11-15, 18-19, 22-28, 30-32, 34-36 and 38-39 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over *Cavanaugh* (US 5,809,507) in view of *Bannon* (US 5,297,279), in further view of *Mattis* (US 6,453,319). The Examiner essentially stated that the combined teachings of *Cavanaugh, Bannon* and *Mattis* teach or suggest all of the limitations of Claims 1, 4-8, 11-15, 18-19, 22-28, 30-32, 34-36 and 38-39.

Claims 1, 15, 26 and 38 are the independent claims.

Claims 1 and 38 claim, and Claim 15 essentially claims, "generating automatically, by the processor, an index to a property of the object if the counter corresponding to the property exceeds a predefined threshold."

Referring to *Cavanaugh*; the Examiner points to col. 17, lines 34-42 of *Cavanaugh* as teaching "generating automatically, by the processor, an index to a property of the object if the counter corresponding to the property exceeds a predefined threshold" (see page 4 of the Office Action). Applicants respectfully disagree.

Briefly, the purpose of the index in *Cavanaugh* is as follows. In *Cavanaugh*, a data store is a data structure into which persistent data (e.g., data objects) is written into memory (see col.

10, lines 34-35). Methods can be invoked to retrieve these data objects from the data store (see col. 9, lines 52-56). Each object is tied to an object identifier (OID), which includes an index (1016) that points to the object's location in the data store (see col. 16, line 66 – col. 17, line 6 and FIG. 10). The purpose of this index is to allow for retrieval of the object from the data store.

More specifically, in *Cavanaugh*, it follows that because every object in a data store is associated with an OID, an index is generated for every object in the data store. Further, generation of the indexes in *Cavanaugh* has no connection to a counter or a predefined threshold. Indeed, because indexes are always generated for every object in a data store, regardless of the number of times the object is accessed, *Cavanaugh* has no need for implementing a counter and a predefined threshold when generating indexes. By way of comparison, in Claims 1, 15 and 38, an index is to a property of an object is only generated upon a counter exceeding a predefined threshold.

In addition to generating an index for every object in a data store, *Cavanaugh* further discloses a use count value (1018) (see col. 17, lines 44-54 and FIG. 10). However, this use count value is different from the counter claimed in Claims 1, 15 and 38. For example, the use count in *Cavanaugh* is utilized to determine the validity of a corresponding index by being incremented each time the index is reused (e.g., an outdated value of a use count signifies that the data store object associated with the index has not been accessed for a long period of time, and typically indicates that the index is invalid). Stated simply, the use count is not utilized in conjunction with a predefined threshold and is not utilized to determine whether to generate an index. In stark contrast, the counter in Claims 1, 15 and 38 is utilized to determine whether to generate an index, and an index is only generated when the counter exceeds a predefined threshold. For example,

unlike *Cavanaugh*, an index in Claims 1, 15 and 38 is not generated for every object existing in a data store. Rather, each time a property of an object is accessed, a counter is incremented, and only when the counter exceeds a predefined threshold is an index generated. This scheme is clearly not taught or suggested by *Cavanaugh*.

Further illustrating the distinction between *Cavanaugh* and Claims 1, 15 and 38, consider that while the use count in *Cavanaugh* is utilized after the index has already been generated in order test the validity of the existing index, the counter in Claims 1, 15 and 38 is utilized before the index is generated in order to determine whether the index should even be generated.

In summation, *Cavanaugh* is deficient in two respects:

- *Cavanaugh* does not disclose utilizing a counter with a predefined threshold in any manner.
- *Cavanaugh* does not disclose generating an index only upon a counter exceeding a predefined threshold.

For at least the foregoing reasons, *Cavanaugh* clearly does not teach or suggest all of the limitations of Claims 1, 15 and 38.

Referring to *Bannon*; *Bannon* does not teach or suggest "generating automatically, by the processor, an index to a property of the object if the counter corresponding to the property exceeds a predefined threshold" as claimed in Claims 1 and 38, and essentially as claimed in Claim 15. In *Bannon*, an object-oriented database (OODB) is utilized to provide long-term storage and retrieval of objects created by application programs written at least in part in object-oriented programming languages (see col. 5, lines 39-45). *Bannon* is silent on utilizing an index

with a counter or a predefined threshold in any manner. Thus, *Bannon* fails to cure the deficiencies of *Cavanaugh*.

Referring to *Mattis*; *Mattis* does not teach or suggest "generating automatically, by the processor, an index to a property of the object if the counter corresponding to the property exceeds a predefined threshold" as claimed in Claims 1 and 38, and essentially as claimed in Claim 15. In *Mattis*, objects on a disk are indexed using keys (see col. 7, lines 44-47). An index/key is created for each object on the disk (see col. 8, lines 28-30). Nowhere does *Mattis* teach or suggest generating an index to a property of an object upon a counter exceeding a predefined threshold, essentially as claimed in Claims 1, 15 and 38. *Mattis* further teaches counters that are used to track the most recently used or least recently used objects on a disk (see col. 25, lines 61-65). The counters are not utilized with a predefined threshold and are not utilized for the purpose of determining whether to generate an index, essentially as claimed in Claims 1, 15 and 38. Thus, *Mattis* fails to cure the deficiencies of *Cavanaugh* and *Bannon*.

The combination of *Cavanaugh*, *Bannon* and *Mattis* teaches an object-oriented database (OODB), indexes to all objects in the OODB, use counts to determine whether the indexes are invalid, and counters to track the most recently used or least recently used objects in the OODB. The combination does not teach or suggest "generating automatically, by the processor, an index to a property of the object if the counter corresponding to the property exceeds a predefined threshold" as claimed in Claims 1 and 38 and essentially as claimed in Claim 15. Accordingly, the combination does not teach or suggest all of the limitations of Claims 1, 15 and 38.

Claim 26 claims, *inter alia*, "a search module for searching the database table for one of the properties of the object, wherein the counter corresponding to the one of the properties is incremented upon searching for the one of the properties; an access history module for monitoring the counter of each property, and for invoking an index creation module upon one of the counters exceeding a predefined threshold, wherein an index to each property is only generated upon the corresponding counter exceeding the predefined threshold" (emphasis added).

Referring to *Cavanaugh*; *Cavanaugh* does not teach or suggest the above-mentioned limitation. As stated above in reference to Claims 1, 15 and 38, indexes in *Cavanaugh* are always generated for every object in a data store, and generation of the indexes has no connection to a counter or a predefined threshold (see col. 16, line 66 – col. 17, line 6 and FIG. 10). By way of comparison, the index creation module in Claim 26 is only invoked upon a counter exceeding a predefined threshold, and an index is only generated for a specific property when the property's counter has exceeded the predefined threshold. *Cavanaugh* does not teach or suggest utilizing a predefined threshold in any manner, much less for the generation of indexes to properties of an object. Further, consider that in Claim 26, creating indexes only for properties that have been accessed a predetermined number of times optimizes access to the persistent storage structure by allowing faster access to frequently used properties. *Cavanaugh*, which teaches indexing every object in a data store, simply does not teach or suggest this optimized scheme. Thus, *Cavanaugh* does not teach or suggest all of the limitations of Claim 26.

Referring to *Bannon*; *Bannon* does not teach or suggest the above-mentioned limitation. In *Bannon*, an object-oriented database (OODB) is utilized to provide long-term storage and

retrieval of objects created by application programs written at least in part in object-oriented programming languages (see col. 5, lines 39-45). *Bannon* is silent on utilizing an index with a counter or a predetermined threshold in any manner. Thus, *Bannon* fails to cure the deficiencies of *Cavanaugh*.

Referring to *Mattis*; *Mattis* does not teach or suggest the above-mentioned limitation. In *Mattis*, objects on a disk are indexed using keys (see col. 7, lines 44-47). An index/key is created for every object on the disk (see col. 8, lines 28-30). Nowhere does *Mattis* teach or suggest invoking an index creation module only upon a counter exceeding a predefined threshold, and generating an index for a specific property only when the property's counter has exceeded the predefined threshold, essentially as claimed in Claim 26. Thus, *Mattis* fails to cure the deficiencies of *Cavanaugh* and *Bannon*.

The combination of *Cavanaugh*, *Bannon* and *Mattis* teaches an object-oriented database (OODB), indexes to all objects in the OODB, use counts to determine whether the indexes are invalid, and counters to track the most recently used or least recently used objects in the OODB. The combination does not teach or suggest "a search module for searching the database table for one of the properties of the object, wherein the counter corresponding to the one of the properties is incremented upon searching for the one of the properties; an access history module for monitoring the counter of each property, and for invoking an index creation module upon one of the counters exceeding a predefined threshold, wherein an index to each property is only generated upon the corresponding counter exceeding the predefined threshold" as claimed in Claim 26. Accordingly, the combination does not teach or suggest all of the limitations of Claim 26.

Therefore, for at least the reasons above, Claims 1, 15, 26 and 38 are believed to be patentable and non-obvious over the combination of *Cavanaugh*, *Bannon* and *Mattis*. Applicants respectfully submit that inasmuch as Claims 4-8, 11-14, 18-19, 22-25, 27-28, 30-32, 34-36 and 39 are dependent on Claims 1, 15, 26 and 38, and Claims 1, 15, 26 and 38 are patentable over the cited references, Claims 4-8, 11-14, 18-19, 22-25, 27-28, 30-32, 34-36 and 39 are allowable as being dependent on allowable independent claims. Withdrawal of the instant rejection is respectfully requested.

CONCLUSION

In view of the foregoing, it is believed that all claims now pending patentability define the subject invention over the prior art of record and are in condition for allowance. Early and favorable reconsideration of the case is respectfully requested.

Respectfully submitted,

Date: June 28, 2010

By: /Nathaniel T. Wallace/
Nathaniel T. Wallace
Reg. No. 48,909
Attorney for Applicant(s)

F. Chau & Associates, LLC
130 Woodbury Road
Woodbury, New York 11797
TEL: (516) 692-8888
FAX: (516) 692-8889